DIP 7th Lecture

October 3, 2023

1 Lecture 7: Digital Image Processing

Neighbourhood Operations

1.0.1 Objectives

- To understand the concept of neighbourhood operations
- To understand the concept of averaging filter

1.0.2 Importing the required libraries

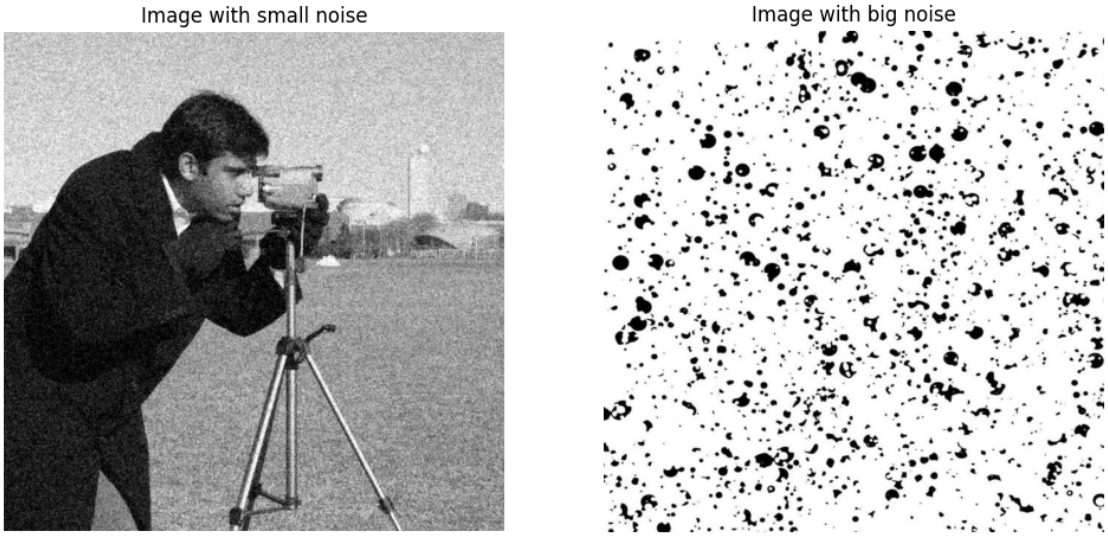
[ ]:
```
import numpy as np
import cv2
import matplotlib.pyplot as plt
import math
```

1.0.3 Reading the Image

Reading the image using the imread() function of the cv2 library and storing it in the variable image. Convert the image to grayscale using the cvtColor() function of the cv2 library.

[ ]:
```
small_noise_img = cv2.imread("small_noise.png")
small_noise_img = cv2.cvtColor(small_noise_img, cv2.COLOR_BGR2GRAY)

big_noise_big = cv2.imread("big_noise.png")
big_noise_big = cv2.cvtColor(big_noise_big, cv2.COLOR_BGR2GRAY)

plt.figure(figsize=(12, 10))

# Plot the image with small noise
plt.subplot(1, 2, 1)
plt.imshow(cv2.cvtColor(small_noise_img, cv2.COLOR_BGR2RGB))  # Convert BGR to␣
 ↪RGB for proper display
plt.title("Image with small noise")
plt.axis("off")

# Plot the image with big noise
```

```
plt.subplot(1, 2, 2)
plt.imshow(big_noise_big, cmap="gray")
plt.title("Image with big noise")
plt.axis("off")

plt.show()
```



1.0.4 3 x 3 Averaging Filter

The averaging filter is a simple and frequently used filter for smoothing images. It is a linear filter that replaces the central pixel with the average of the pixels in the surrounding neighbourhood. The averaging filter is also called the low-pass filter because it passes the low-frequency components and attenuates the high-frequency components. The averaging filter is given by the following equation:

$$g(x,y) = \frac{1}{9} \sum_{s=-1}^{1} \sum_{t=-1}^{1} f(x+s, y+t)$$

where, $f(x,y)$ is the input image, $g(x,y)$ is the output image, and s and t are the neighbourhood indices.

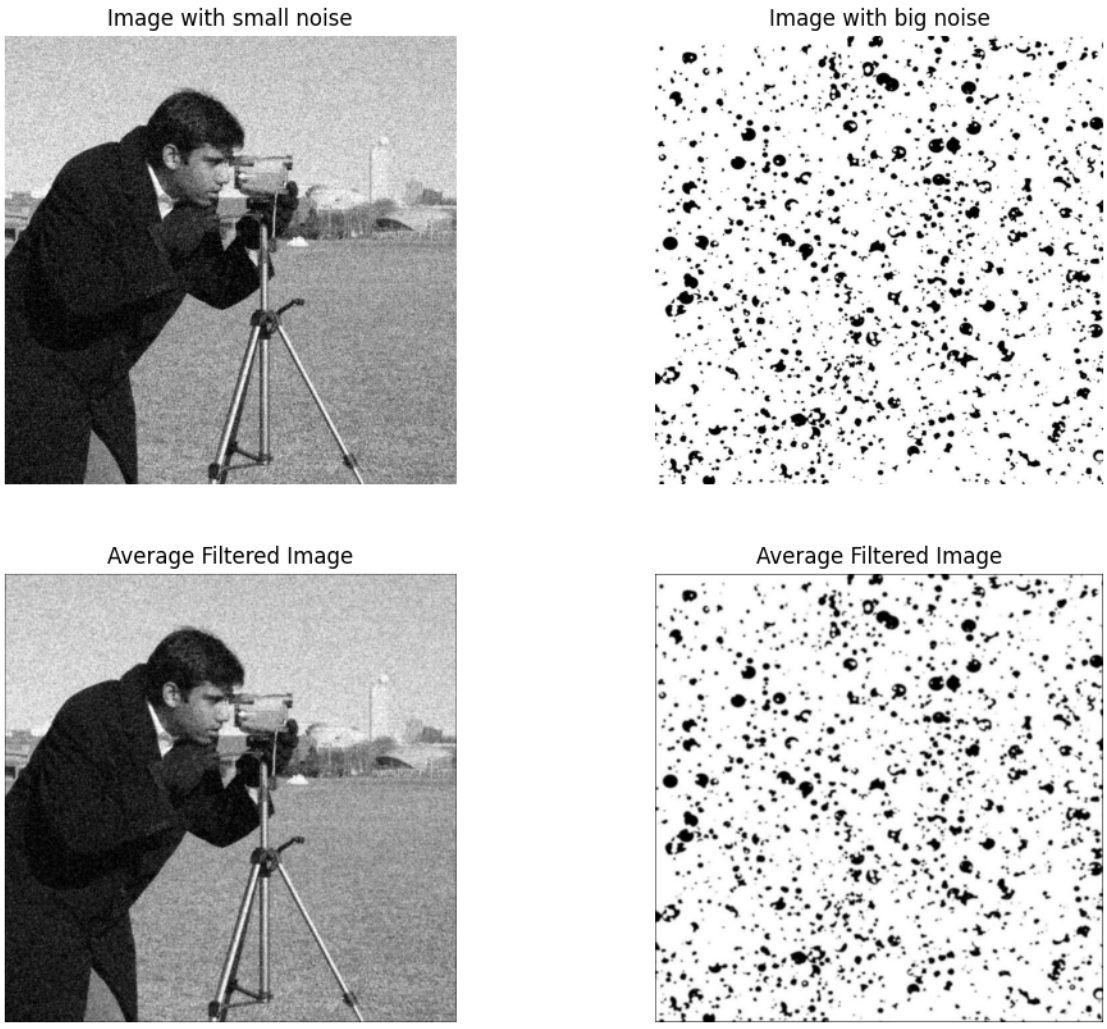
Write a function averaging_filter() that takes the image and returns the filtered image.

[ ]:
```
def average_filter(image):
    """
    This function will perform average filtering on the image.
    :param image: The image to be filtered.
    :param kernel_size: The size of the kernel.
    :return: The filtered image.
    """
```

```
    # Get the image height and width
    height, width = image.shape

    # Create a numpy array to hold the filtered image
    filtered_image = np.zeros_like(image)

    for i in range(1, height - 1):
        for j in range(1, width - 1):
            # Get the 3x3 neighborhood
            neighborhood = image[i - 1 : i + 2, j - 1 : j + 2]

            # Apply the averaging filter and save the value
            filtered_image[i, j] = np.mean(neighborhood)

    return filtered_image
```

[ ]:
```
filter_small_noise = average_filter(small_noise_img)
cv2.imwrite("filtered_small_noise.png", filter_small_noise)

filter_big_noise = average_filter(big_noise_big)
cv2.imwrite("filtered_big_noise.png", filter_big_noise)

plt.figure(figsize=(12, 10))

# Plot the image with small noise
plt.subplot(2, 2, 1)
plt.imshow(cv2.cvtColor(small_noise_img, cv2.COLOR_BGR2RGB))  # Convert BGR to␣
 ↪RGB for proper display
plt.title("Image with small noise")
plt.axis("off")

# Plot the image with big noise
plt.subplot(2, 2, 2)
plt.imshow(big_noise_big, cmap="gray")
plt.title("Image with big noise")
plt.axis("off")


# Plot filtered the image with small noise
plt.subplot(2, 2, 3)
plt.imshow(filter_small_noise, cmap="gray")
plt.title("Average Filtered Image")
plt.axis("off")

# Plot filtered the image with big noise
plt.subplot(2, 2, 4)
plt.imshow(filter_big_noise, cmap="gray")
```

```
plt.title("Average Filtered Image")
plt.axis("off")

plt.show()
```



1.0.5 Conclusion

In this experiment, we learnt to perform neighbourhood operations on an image. We also learnt to perform averaging filter on an image.